AMENDMENTS TO THE SPECIFICATION

Please replace the paragraph beginning on Line 2, Page 13 with the following rewritten paragraph:

A complex application 108 that is being ported to IBM OS/390 UNIX, such as a database, may incorporate at least three types of I/O access code. The I/O access code may be unmodified and the application 108 may rely on the facilities of a computer system 101, such as an operating system 110. Alternatively, the application 108 may modify the I/O access code for a computer system 101 environment that only supports queued I/O access, such as IBM OS/390 UNIX. Finally, the application 108 may augment general-purpose I/O access operations with customized direct I/O access operations that take advantage of any I/O interfaces that may be provided by the computer system 101 and that may enhance I/O performance of the application 108. It is assumed that a complex application 108 manages serialization of I/O requests 121 in a thread-safe manner. The present invention optimizes I/O requests121 for complex applications 108 that have already taken on the burden of managing serialization and have, for the files 102 associated with these I/O requests 121, abandoned the I/O caching scheme typically available via the general-purpose file system 115. Such a general-purpose file system 115 may be optimized to support queued I/O. Those skilled in the art will appreciate that applications 108 developed for operation on other operating systems 110 which support application-level I/O caching and that use direct, asynchronous I/O, will incorporate program code that performs such serialization. More particularly, the present invention may augment facilities on the IBM OS/390, such as the

high-level language APIs 116, so that an application that is ported to the IBM OS/390 UNIX System Services will operate more efficiently. OS/390 UNIX provides support for APIs 116 and an interactive shell interface 126. The terms "I/O request" and "I/O command" will be used interchangeably herein.

Please replace the paragraph beginning on Line 3, Page 18 with the following rewritten paragraph:

Direct I/O requests 121 issued by the application 108 outside of the I/O programmatic loops 171 are passed to the low-level direct I/O interface 113 immediately and without aggregation. I/O programmatic loops 140 must be selected such that queued I/O requests for a performance file ~~370~~ 170 are not included in the I/O programmatic loop 140. Outside of the I/O programmatic loops ~~302~~ 171 and for a given performance file 208, the performance file 208 may be opened or closed, using standard operating system APIs 116, for specific types of I/O commands 121. Recall that OS/390 UNIX provides support for APIs 116 and an interactive shell interface 126. In particular, a performance file 208 is often initially opened for queued I/O. Then, following the issuance of a number of queued I/O commands 121, the performance file 208 may be closed for queued I/O, then reopened for direct I/O processing. When a performance file 208 is open for queued I/O, the preferred embodiment of the present invention receives queued I/O commands for a performance file 170, translates them to direct I/O commands 121 that are appropriate for the low-level direct I/O interface 113, passes the translated requests to the low-level direct I/O interface 113, and then waits for the direct I/O commands 121 to complete.

Please replace the paragraph beginning on Line 18, Page 19 with the following rewritten paragraph:

The high-performance improvement code module 112 operates at the behest of the system-dependent code module 111 to translate I/O requests 121 for the low-level direct I/O interface 113. The high-performance improvement code module 112 includes API features 116 to facilitate its use. It will be noted that I/O requests 121 that are not identified for translation by the high-performance improvement code module 112 are passed from the application 108 to the system-dependent code module 111 and on to the operating system 110 without modification. More specifically, the file system 115 then translates the I/O requests 121 so that they may be passed on to the low-level direct I/O interface 113. The low-level direct I/O interface 113 then passes the I/O requests 121 to the I/O subsystem 114 for transmission to the disk 122.